CMPS 1600 Introduction to Computer Science II – Spring 14

2/12/14

4. Homework

Programming portion due **Thursday 2/20/14** at 11:55pm on Blackboard.
Written portion (problems 1(b),(g)) due the next day at the beginning of class.

Please zip the project directory for this homework, and use the following naming convention for the name of the project (and directory): `lastName_firstName_hw4`
In order to receive any credit for the programming portions, you are required to thoroughly comment your code.

1. **Collection (20 points)**
   Consider the `Collection` interface developed in class:

```
public interface Collection<T>{
    public void add(T item);     // Adds one item
    public void remove(T item);  // Removes ALL elements equal to item.
    public boolean contains(T item); // True if item is in collection.
}
```

   Remember, that `remove` removes all elements equal to `item`, and that one uses the `equals` method to compare two items (why??).

   (a) (2 point) Give the runtimes for all the methods in the class `ArrayBag` that was developed in class. Assume n is the number of elements stored in the bag. Justify your answers shortly. (You can add the runtimes in comments in the code.)

   (b) (1 point) The runtime for `remove` in `ArrayBag` is somewhat slow. Do you have ideas how you could speed it up? What would be the runtime? Describe your idea in words (code is not necessary).

   (c) (7 points) Implement a class `LinkedBag` that implements the `Collection` interface using a linked list. Duplicate elements are allowed, and `LinkedBag` should not have any other public methods than the ones in the `Collection` interface and the constructor. Make sure that you use the proper access modifiers, and add the runtimes of the methods in comments to your code. (For the `remove` method, make sure to handle special cases such as removing the first node.)

   (d) (3 points) Implement a class `SpecialLinkedBag` that `extends LinkedBag`, and adds the additional functionality of a method `addAll` which takes a `T[]` array as argument and adds all its non-`null` elements to the bag.

FLIP OVER TO BACK PAGE ⟹

(e) (3 points) In the `main` method of a separate tester class, test all methods of `LinkedBag<String>` for elements of type `String`. (Don't test the method `addAll` yet.) Now, instead of using a bag variable of type `LinkedBag<String>` your code should work just the same if you change the bag variable to type `ArrayBag<String>` (why does this work?). In fact, there are eight different ways you can declare the bag variable to work with the exact same test code. (This is because `LinkedBag` implements `Collection` and is extended by `SpecialLinkedBag`, and `ArrayBag` also implements `Collection`.) In comments, give all eight possible declarations and instantiations for the bag variable that will work with the exact same test code.

(f) (2 points) Now test all methods of `SpecialLinkedBag<String>`. Which of the eight declarations and instantiations for the bag variable work for this test code? Justify your answer.

(g) (2 point) Draw a UML class diagram depicting all classes used in your project and the relationships between them.

(h) **Extra credit (2 points):** For two objects `a` and `b` of the same type, what does `a==b` mean (think about what happens in memory...) and what does `a.equals(b)` mean? For example, if `a` and `b` are objects of type `Point`, when is `a==b` true and when is it false? Give an implementation of the `equals` method for `Point` such that `a.equals(b)` is true for "equal" points. (Do not consider `String` objects for this, because Java treats them similar to primitive types.)